第 2 に，人間がもつ不完全性や非合理性というものに注目してみよう．現在の危機的状況をうみだすもとになっているひとつの問題点は，従来のプログラムやプログラマのモデルにあるとかんがえられる．従来のプログラム理論においては，プログラムは合理的かつ完全であることがもとめられる．プログラムは正当か不正かのいずれかであり，その中間はない．これは，すなわちプログラマが合理的かつ完全であることをもとめていることになる．ところが，プログラマは人間であり，したがって非合理かつ不完全である．ソフトウェア開発の現場をみれば，プログラマを合理的かつ完全な存在とみることこそ不合理であることはあきらかであろう [Nis 88, Nis 90]．

このような視点からすると，われわれは不完全で非合理な人間がつくる不完全で非合理なプログラムに関する理論をつくらなければならないのではないだろうか．「人間的」なプログラミングの理論においては，些細なバグによって致命的な結果がもたらされるようなことはなく，完全でないプログラムもそれなりに動作することが保証されるべきであろう．

非合理性のとりこみという点でも，第 1 の視点からとりあげたいくつかの関連研究は興味ぶかい．ニューラル・ネットは "非合理" な入力を処理できるし，GA はその計算機構じたいに非合理性がふくまれているようにおもわれる[注1]．しかし，非合理性に関しては他の面からの研究も必要だとおもわれる．

第 3 に，システムがもつ非還元性 (全体性) に注目してみよう．従来のソフトウェア開発法はシステムを独立なモジュールに分割するという還元論的な方法 (分割統治法) であり，そこからはみだしたものはもはや系統的にあつかえない．しかし，現在の大規模システムや AI システムなどにおいては，現実には非還元論的なソフトウェアが構築されているとかんがえられる [Nis 88]．モジュール性をくずす大域変数をプログラムから排除できないのはそのためではないだろうか．AI においても非還元的なシステムの適当な (汎用性がある) モデルはまだ存在しないとかんがえられる．

このような状況のもとで，われわれは自己組織系の記述をめざし，人間のもつ不完全性や非合理性までをもとりこみ，かつ数理解析が可能とかんがえられるモデル CPM を考案した．2 章では自己組織系とはなにかという問題について考察し，その必要条件とかんがえられるものをしめす．3 章では CPM を定義し，それに関して 2 章の条件を検討し，例題をしめす．4 章では CPM にもとづくプログラムのふるまいを分析する．5 章では例題に関する実験結果をしめす．

2. 自己組織系とはなにか？

コンピュータ・システムを構築するときに手本となるのは，自然システムすなわち自然がつくりあげたシステムや自然につくりあげられた社会システムである．自然システムの特徴として，システム哲学者 Laszlo はつぎのような項目をあげている[Las 72][注2]．

(1) 全体性 :「自然システムは非還元的特性をもった全体である」．

(2) 自己安定性 :「自然システムは変化する環境のなかで自らを存続させる」(負のフィードバック)．

(3) 自己組織性 :「 自然システムは環境の挑戦に呼応して自らを創造する．すなわち，環境の変化に応じて，あらたな安定状態すなわち秩序をもとめて創造的に適応する」(正のフィードバック)．

(4) 階層性 :「自然システムは自然の階層性のなかで相互に触れ合いながら整序しあっている」．

これらの特徴のうち (2), (3) はそのシステムが環境とのやりとりをする，すなわち開放性をもっていることを前提としている．自己組織系とは自己組織性をもったシステムのことだが，自己組織系はそれだけでなく前記の 4 つのすべての特徴をそなえたシステムのことをいうのだとかんがえられる．

自己組織系を工学的につくりだすためには，まずその性質をより明確にし，モデルをつくる必要があるとかんがえられる[注3]．そこで，まず自己組織系がみたすべき必要条件についてかんがえよう[注4]．

[注1] コーディングがよほどうまくおこなわれたばあい以外は，GA におけるくみかえや突然変異という操作に合理的根拠があるとはかんがえられないからである．

[注2] ただし，「 」内以外でのことばのつかいかたはかならずしも Laszlo によるものではない．

[注3] 自己組織系を "工学的に" つくりだそうというかんがえじたいに疑問がないわけではない．なぜなら還元主義に毒された従来の工学的な方法を適用したとたんに自己組織化はわれわれの手からにげていってしまいかねないからである．しかし，とりあえずこの疑問には目をつぶる．

[注4] ただし，これらは自己組織系の必要条件ではあっても十分条件ではない．必要十分条件をしめすこと，すなわち自己組織系を定義することは，すくなくともいまのところはできないとかんがえられる．いいかえれば，それをいま無理に定義すると，Jantsch [Jan 80] らがつかっているこのことばの意味からはずれてしまうとかんがえられる．

(1) 組織化条件 : 秩序の生成

自己組織系は組織化をおこなう．すなわち，秩序をうみだす[注5]．

システムが秩序をうみだしているかどうかを検証するには，秩序の指標が必要であろう．指標としてすぐにおもいつくのはエントロピーである．しかし，第 1 に，どのようなシステムにもあてはまるエントロピーの定義が存在するわけではない．第 2 に，エントロピーが秩序の指標としてかならずしも適切なものではないことが指摘されている [Hak 78, p. 15] [Deg 88]．しかし，エントロピー以外の指標をかんがえると，さらに客観性はうすれてしまう[注6]．したがって，すくなくとも現在のところ，秩序ということばの意味を一般的に，また完全に客観的に定義することはできず，秩序の指標は対象となるシステムごとに定義するほかはないとかんがえられる．

ところで (1) の条件をみたすものは "組織化" ではあるが，"自己組織化" であるかどうかはこれだけではきまらない．そこで，つぎの条件をくわえる．

(2) 自発性条件 : 非決定性の存在

自己組織化であるためには，システムが外部からのちからによって決定論的に (あるいは運命論的に) 動作するのではなく，自発的に動作する (ようにみえる) ことが必要だとかんがえられる[注7]．したがって，自己組織系を外部から観察すれば，非決定論的に動作しているようにみえるはずである．すなわち，自己組織系を構成する機能要素は自由をもっていて，非決定的な選択をおこなうことによってシステムのつぎの状態をきめていく．

ここで非決定性とは，ある初期状態にあるシステムが，そのシステムを記述するパラメタだけでは，いくつかの可能性のうちのどの状態をとるかがきまらないという性質のことである．

ところで，システム要素の自発的な選択によって動作しているから，トップ・ダウンな要素もあるものの，基本的には自己組織化はボトム・アップなはたらきによってもたらされるといえるだろう．

システムのある時刻の状態がそのシステムのそれ以前の状態からきまるとき，それは動力学系 (dynamical system) とよばれる[注8]．動力学系が安定状態の近傍にあるときは，撹乱をうけてもそのシステムをその安定状態にたもとうとするはたらき，すなわち自己安定性がはたらく．しかし，システムがある臨界点をこえ，安定状態を維持できない，またはそれが不利な状況がおこると，他の安定状態にうつる．

この移動には上記の非決定性がはたらくであろう．すなわち，つぎの安定状態にいたるシステムの軌道が分岐していて，どちらの軌道が選択されるかはゆらぎやランダムネスやその他の微小な原因によって左右される (これらの原因に対する正のフィードバックがはたらく) ため，あらかじめ予測できない [Pri 87 など][注9, 注10]．このような現象は，物理系などにおいては "対称性のやぶれ" とよばれる [Pri 84]．分岐は，おおくのばあい，図 2.1 のようにカスケードをなしている．このような軌道の選択は，非決定的であるために自発的であるようにみえる．このように要素が自発的に動作するということは，システムが機械論的ではなく有機論的であることを意味する．

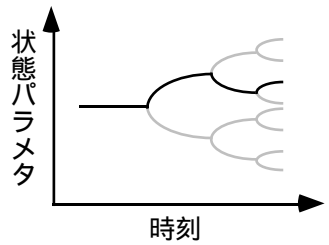


図 2.1　システムの軌道のカスケード状の分岐

自己組織系がもつ非決定性は，ある意味では "非合理性" とよべるであろう．逆にいえば，1 章でのべたような人間がもつ非合理性をこの非決定性に吸収できるのではないかともかんがえられる．この点に関しては 4 章において論じる．

(3) コヒーレント性条件

システムの性質を維持しつつ変化していくシステムはコヒーレントなシステム [Jan 80] とよばれる．自己組織化はコヒーレントなシステムで生じる現象であるという点は，さまざまな自己組織化に共

[注5] 秩序をうみだすのが自己組織系だといってよいかどうかは疑問もある．なぜなら，結晶がもつような静的な秩序と，生物のような自己組織系がうみだす動的な秩序とのあいだにはおおきなへだたりがあるからである．

[注6] Haken らによるシナジェティクスにおいては，秩序の種類と程度をあらわす量として "秩序パラメタ" がつかわれる．秩序パラメタは対象となる系ごとに定義される．

[注7] この条件は，自己組織系である人間のわれわれにとって，感覚的にも納得がいくものであるようにおもわれる．

[注8] 通常は決定論的なシステムだけを動力学系とよぶが，ここではシステムの状態が非決定的に (確率的に) きまるシステム (確率的動力学系) をかんがえる．

[注9] 軌道の選択においてはそれにからむ各種の要因が競合するが，ひとつの軌道の選択後は，それらは協調するであろう．このような競合と協調も自己組織系を特徴づける．

[注10] あらたな安定状態じたいがあらかじめ存在せず，系がみずからつくりだすばあいもあるとかんがえられる．

通の特徴だとかんがえられる．

ところで，Prigogine や Haken らによって研究されてきた物理・化学系においては，自己組織化は非平衡非線形の開放系で生じるという重要な特徴があった．しかし，情報系における自己組織化がこれらのシステムとどのような関係があるかは，いまのところあきらかではない．また，自己組織性に関しては自己参照 (self-reference) またはリフレクションの問題が重要だとかんがえられている (たとえば [Jan 80]) が，この報告ではそれらについてはふれない．

3. 自己組織のための計算モデル CPM

この章では，前章の分析結果にもとづいて，自己組織系を記述するための計算モデルである化学的プログラミング・モデル (Chemical Programming Model) を提案する．3.2 節で CPM の詳細をしめすが，そのまえに 3.1 節でより一般的な自己組織系のモデルをしめす．また 3.3 節では，2 章でしめした自己組織系の必要条件が CPM においてどのようにすればみたされるかをかんがえ，3.4 節では例題として N クウィーン問題をとりあげる．

3.1 自己組織系の基本モデル

自己組織系の基本モデルとして図 3.1 のようなものをかんがえることができる．このモデルがあらゆる自己組織系にあてはまると主張することはできないだろう[注11]が，われわれが対象としようとしている情報の自己組織をおこなうシステムをはじめとして，散逸構造をうみだす熱力学系やその他のさまざまな自己組織系のモデルとなっているとかんがえられる．

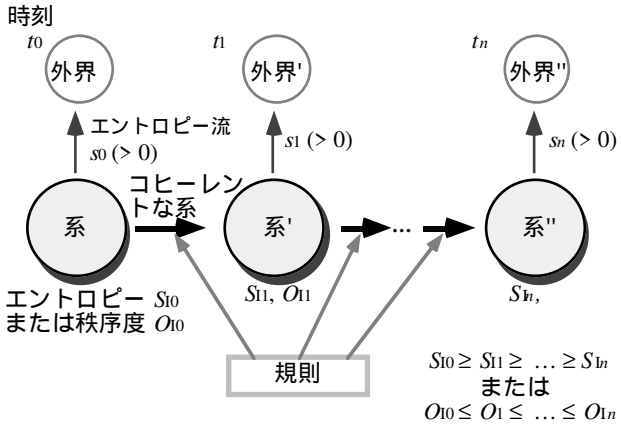


図 3.1　自己組織系の基本モデル

このモデルにおいて，システムは時間とともに変化する (したがってそれは動力学系として表現できる)．入力がなければシステムは通常，一定時間後には定常状態に達している．しかし，そのばあいでもあとから入力があれば再度変化しはじめるという散逸構造的な性質がある．この性質を漸進性とよぶ．

システムに対して，秩序の指標としてのエントロピーまたは (大域) 秩序度が定義されている．エントロピーが定義されているばあいには，それは時間とともに減少する．秩序度が定義されたばあいには，それは時間とともに増加する．熱力学系のばあいには，熱力学第 2 法則をみたしつつエントロピーを減少させるためには，エントロピーを外界にすてなければならない．したがって，システムは開放系でなければならない．このような法則が存在しないシステムにおいては，(この要請だけをかんがえるかぎりは) かならずしも開放系でなくてもよいであろう．

また，システムの変化のしかたを支配する規則あるいは法則が存在するはずである[注12]．その規則の記述形式としてはさまざまなかたちがかんがえられる．

上記のモデルをより具体化して，情報の自己組織化のためのモデルを構成しよう．このモデルにおける規則の記述のために (化学反応式のような，あるいはプロダクション・システムにおけるような) プロダクション規則を採用する．その理由は次のとおりである．

(1) プロダクション・システムはボトム・アップな計算のモデルとして適当だとかんがえられる

規則ベースの計算モデルとして，前向き推論のシステムすなわち OPS5 のようなプロダクション・システムと，後向き推論のシステムすなわち Prolog のような論理型言語，あるいは後向きプロダクション・システムとがある[注13]．後者はトップ・ダウンな計算機構であるのに対して，前者はボトム・アップな計算機構である．2 章でのべたように自己組織化は基本的にボトム・アップなプロセスであり，そのモデルとしては前者のほうが適しているとかんがえられる．したがって，前向き推論のプロダクション・システムを採用する．

[注11] 自己組織系の全体像がみえていない現在，あらゆる自己組織系を記述するベースになりうるほど一般性があるモデルをつくることは不可能だとかんがえられる．

[注12] この報告ではこの規則をあまくだりのものとするが，本来それは自己参照によって変化するものであり，また "規則" という結晶したかたちで記述されるとはかぎらない．さらに，4 章でのべるのように，この規則によってシステムのふるまいが完全に決定論的にきめられるのではなく，システムの構成要素には自由がのこされる．

[注13] 図 3.1 における規則の表現としてかならずしもプログラミング言語論でいう規則ベースの表現をとる必然性はないが，後述する数理解析の容易さなどをかんがえると規則ベースの表現がよいとかんがえられる．

(2) 化学反応系とのアナロジーをつかう

情報の自己組織系を構成するための方法論はまだまったくえられていないといってよい．したがって，それを実現するためには，他の分野からのアナロジーをつかうのがよいとかんがえられる．散逸構造理論の舞台である化学反応系は，この目的に適しているとかんがえられる．

(3) プロダクション・システムは自由度のたかいプログラムを記述しやすい

手続き型言語はもちろん，関数型言語や論理型言語においても実行順序がきっちりと (半順序として) きめられていて自由度がすくない．それに対してプロダクション・システムは，規則が適用される順序に関して自由度がたかい[注14]．

また，システムは空間的にも時間的にも離散的だとする．空間的に離散的であるということは，これから定義しようとしているモデルが記号の自己組織化のためのモデルであるというところからくる．なぜなら，言語学・記号論において Saussure [Sau 15] 以来指摘されてきているように，人間があつかう記号は離散的だとかんがえられるからである．

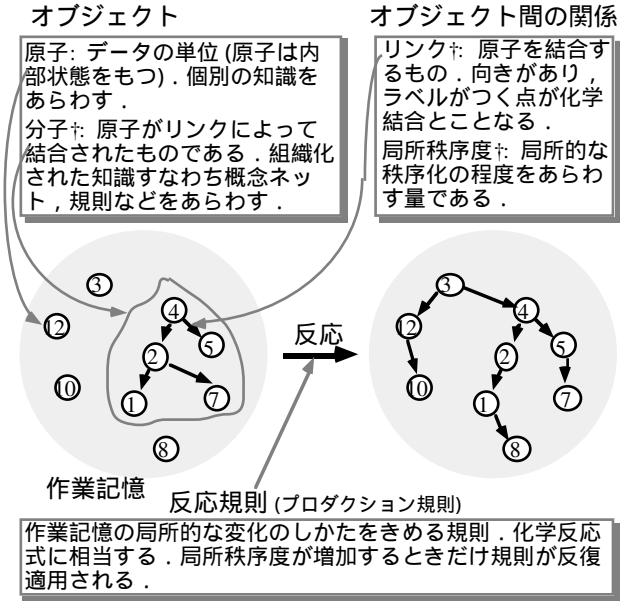
3.2 化学的プログラミング・モデル (CPM)

自己組織系の記述をめざした計算モデルとして CPM を定義する．CPM の構成要素を図 3.2 にしめす．プログラムの作用対象であるデータ集合は，通常のプロダクション・システムにおけるのと同様に作業記憶とよぶ．作業記憶がふくむ単位的なオブジェクトは原子とよぶ．原子は内部状態をもち，他の原子へのリンクをもつことができる[注15]．リンクによって結合された原子の集合を分子とよぶ．原子とリンクとによって，任意のリスト，木，グラフ，ネットワークなどの離散的な構造を表現できる．

システムの状態を変化させるのは反応規則である．反応規則はプロダクション・システムにおけるプロダクション規則のかたちで記述される．これは化学反応における反応式に似たものだといえる．すなわち， 反応規則はつぎのようなかたちをしている．

LHS → RHS.

左辺 LHS，右辺 RHS には，ともに原子または分子のパタンの列が記述される．左辺にマッチする原子・分子が存在するときに規則は動作可能になる．規則が動作すると，左辺に記述された原子・分子は消滅し，右辺に記述された原子・分子が生成される．たとえば図 3.3 の例では“酸素分子” と“水素分子” が反応によって消滅し，“水分子” が生成される[注16,注17]．



† OPS5 などの従来のプロダクション・システム記述用言語にはない特徴的機能．

図 3.2　CPM (化学的プログラミング・モデル)

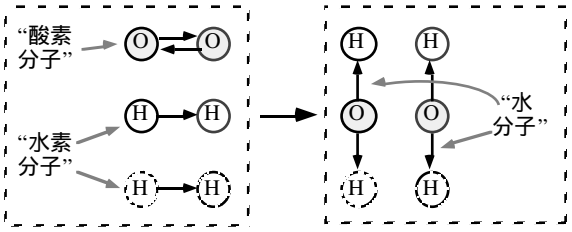


図 3.3　反応規則の例

ただし，反応規則の左辺にマッチする原子・分子が存在すればただちに規則が適用されるわけではない．規則が適用されるためには，局所秩序度に関する条件がみたされなければならない．ここで局所秩序度とは局所的な秩序の指標であり，通常は整数値

[注14] この性質は (1) とも関係があるとかんがえられる．

[注15] リンクによってデータ間の関係を陽にあつかえるという点が，従来のプロダクション・システムとくらべての CPM のひとつの特徴だとかんがえられる．また，GA においては直線的なデータ構造しかゆるされないのに対して，CPM においてはリンクによって任意のネットワーク構造があらわせることも指摘しておきたい．

[注16] ただし，この規則は水を生成する化学反応のモデルとしては適当だといえない．すなわち，意味のない例である．

[注17] 反応規則はこのように視覚的に表現するのが便利である．なぜなら，通常のプロダクション・システムにおける規則とちがってリンクが存在し，それをつかってデータ構造が表現されるからである．

か実数値をとる[注18]．局所秩序度はつぎの 2 つのうちのいずれかのかたちで定義される．

(1) 自己秩序度 *o*(*e*)

1 個の原子 *e* に対して定義される．

(2) 相互秩序度 *o*(*e1*, *e2*)

2 個の原子 *e1*, *e2* のあいだに定義される．

規則は，その適用対象の原子の局所秩序度の和が規則適用によって増加するときだけ (あるいは減少しないときだけ) 適用される[注19]．自己秩序度のばあいは，規則の両辺にあらわれるすくなくともひとつのパタンとマッチする原子すべてについての，規則の適用前と適用後の局所秩序度の和を比較して，規則を適用するかどうかをきめる．相互秩序度のばあいは，それらの原子のなかから 2 個をえらんで計算した秩序度のすべてのくみあわせについての和を同様に比較して，規則を適用するかどうかをきめる．

ところで，上記の局所秩序度を作業記憶全体にわたってたしあわせたものが大域秩序度になるべきである．あたえられた局所秩序度の定義から大域秩序度をこのようにして定義することもできるし，大域秩序度がプログラムの仕様としてあたえられたときに，それから局所秩序度をもとめるばあいにもこの関係がつかえるであろう．

規則の適用にあたって大域的な秩序を (直接) 考慮しないのは，計算が局所的におこなわれるようにするためである．それは効率をよくするという実用的な目的のためでもあり，化学反応系などとのアナロジーのためでもある[注20]．このように局所的な秩序の指標をつかって大域的な秩序をきずくことをめざしている点が，CPM のおおきな特徴のひとつである[注21]．

これ以降，プロダクション規則と，その左辺にマッチしうるデータとの組のことを，プロダクション・システムの用語にならって "インスタンス" とよぶ．ただし，従来のプロダクション・システムとはちがって CPM においては競合集合をつくらない方法を基本とする (5 章参照) ので，インスタンスはシステムによって実際に構成される競合集合の要素ではなく，むしろ仮想的なものである．また，このインスタンスをオブジェクト指向におけるインスタンスと混同しないようにされたい．

CPM においては，上記のような規則の適用が可能であるかぎり，何度でも規則の適用が反復される．そして，規則の適用によって秩序度が増加するインスタンスが存在しない状態にいたると，停止する[注22]．

ところで，局所秩序度だけではプログラムの動作が一意にきまらないばあいがある．すなわち，ある時刻において適用 (発火) 可能なインスタンスは複数個存在しうる．これらのうちのいずれをさきに発火させるか，あるいは並列に発火させるかは，仕様としては定義しない[注23]．ただし，ひとつの原子をかきかえる複数のインスタンスは同時に発火させないものとする[注24]．したがって，CPM にしたがってうまく記述されたプログラムにおいては，秩序度関数によって定義された意味での秩序が，非決定的に，したがって "自己組織" 的に実現されることになる．

3.3 自己組織系の必要条件と CPM

CPM において，2 章でしめした自己組織系の必要条件はすでにみたされているか，あるいはどのようにすればみたされるかをかんがえよう．

(1) 組織化条件 : 秩序の生成

プログラムは局所的な秩序指標がたかまる方向に動作する．したがって，局所的な秩序がすなわち大域的な秩序につながるばあいには，ただちに組織化条件がみたされる．しかし，局所的な秩序が他の部分の局所的な秩序や大域的な秩序と競合的な関係にあるばあいもあるから，ただちに組織

[注18] ここでエントロピーということばをつかわないのは，この用語を理論的に定義されるべき量のためにとっておきたいからである．秩序度は実践的に定義される量である．

[注19] 局所秩序度が変化しないときにも規則を適用するべきかどうかは，プログラムごとに選択できるようにすることがのぞましいとかんがえられる．

[注20] ただし，化学反応系とはちがって CPM にはいまのところ距離の概念をもちこんでいない．したがって，"局所性" の意味は両者においてことなっている．

[注21] いうまでもないが，ゲームにおける探索や遺伝的アルゴリズムにおいては，評価関数として通常は大域的な状態を評価する関数を使用する．CPM による探索がこれらとことなる点のひとつが，この秩序度関数の局所性である．

[注22] ただし，3.1 節でものべたように，その後の入力によってシステムを漸進的に動作させることができる．すなわち，適用可能な規則がなくなるとシステムはユーザ入力まち状態になり，ユーザが終了コマンドを入力しないかぎりはつぎの入力によってふたたび動作する．このようなシステムのつくりは，従来のプロダクション・システムにはなかった特徴的なものだとかんがえられる．

[注23] インスタンスの選択に意識的に非決定性をもちこもうとしている点が，CPM が通常のプロダクション・システムとことなる点のひとつだとかんがえられる．

[注24] このように，並列発火をさまたげないがひとつの原子の並列かきかえはゆるさないという制約は，化学反応におけるのとおなじだとかんがえられる．

化条件がみたされるとはいえない．

このような競合が存在しないばあいには，プログラムの動作は直線的であり，自己組織的とはいえず，あまり興味をひかない．競合が存在するプログラムの例は 3.4 節でしめす．

(2) 自発性条件 : 非決定性の存在

前節でのべたように，同時に発火可能なインスタンスのうちのいずれをさきに発火させるか，あるいは並列に発火させるかは非決定的である．

この非決定性が，自己組織を可能にするうえで非常に重要だとかんがえられる．したがって，今後 CPM を詳細化していくときに，それをどのようにあつかうかが主要な問題になるとかんがえられる．この点に関する課題については 4 章において論じる．

(3) コヒーレント性条件

秩序度をたかめる方向に動作するプロダクション・システムという機構じたいがコヒーレントなものだとかんがえられる．

3.1 節でのべた動作の漸進性は，コヒーレント性と関係がふかいとかんがえられる．

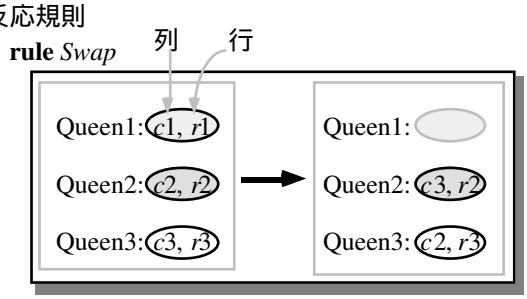
以上で計算モデル CPM のおおすじが定義され，それが自己組織系の必要条件をみたせることがしめされた．ただし，ここでことわっておくが，CPM は自己組織系の記述のためのベースにすぎず，それじたいが自己組織系だとはいえない．すなわち，自己組織系の本質は CPM のなかにではなく，そのうえに記述されるプログラムに存在するはずである．

3.4 例題 : *N* クウィーン問題

CPM によるプログラムの例として *N* クウィーン問題のプログラムをしめす．図 3.4 は解を 1 個だけもとめるプログラムである[注25]．このプログラムはただひとつの規則と，クウィーンの相互秩序度の定義とから構成されている．この規則は，図 3.5 にしめすように 2 個のクウィーン (図 3.4 における Queen2 と Queen3) の列を交換する規則である[注26, 注27]．Queen1 は交換の動作には直接は関係ない．Queen1 のやくわりについては後述する．このプログラムにおける局所秩序度は，2 個のクウィーンが対角方向になければたかく (1 であり)，対角方向にあればひくい (0 である) と定義されている．適当な初期値をあたえて (たとえば全クウィーンを対角線上において) このプログラムを実行させると，適当な 3 個のクウィーンを選択して規則を適用するという動作をくりかえし，結果として *N* クウィーン問題の解がえられると停止する[注28]．

■ 反応規則

rule *Swap*

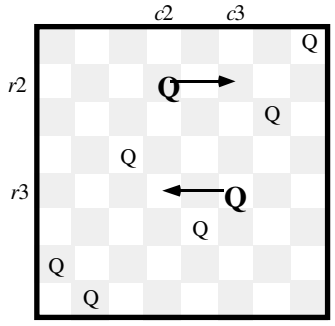


■ 局所秩序度 (相互秩序度)

2 個のクウィーンのあいだで秩序度を定義する。

$$o(x, y) = 0 \quad \textbf{if } x.column - y.column = x.row - y.row \textbf{ or } x.column - y.column = y.row - x.row,$$
$$1 \quad \textbf{otherwise}.$$

図 3.4 CPM による *N* クウィーン問題のプログラム



2 個のクウィーンをえらんでその欄を交換するという操作をくりかえすことによって解に到達することをめざす。

図 3.5 *N* クウィーン問題のプログラムの規則の意味

規則の適用について，図 3.6 を使用してよりくわしく説明する．プログラムを実行させると，処理系は適当なインスタンスをえらんで発火させる．規則は 1 個しかないので，ここでの処理系の仕事は 3 個のクウィーンを選択することだけである．えらばれた 3 個のクウィーンに関して，その規則適用前と適用後における局所秩序度を計算する．この規則のばあい，交換する 2 個のクウィーン Queen2 と Queen3 とのあいだの相互秩序度は変化しない．したがって，Queen2, Queen3 と Queen1 とのあいだの相互秩序度だ

[注25] CPM の性質上，全解探索は困難である．

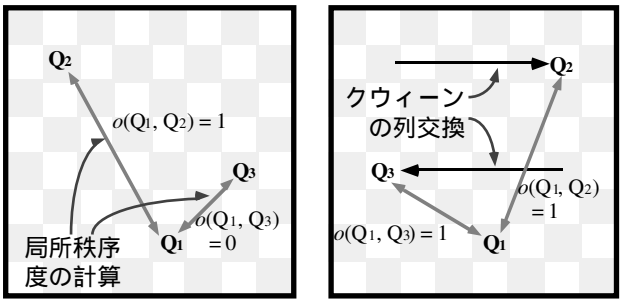
[注26] ただし，実験に使用した SOOP はテキスト・ベースの言語であり，図 3.5 のような視覚言語ではない．

[注27] クウィーンの列の交換によって解をもとめようとしている点で，この解法は清水らによる解法 [Shi 90] にちかい．ただし，清水らは交換の際に他のすべてのクウィーンとの関係をしらべているのに対して，ここでの解法においては他のクウィーンのうちの 1 個だけを考慮している．

[注28] *N* クウィーン問題にかぎらずこのような単純な問題においては，適当な初期状態をあらかじめ設定しておきさえすれば，CPM にしたがうただ 1 個の規則と秩序度関数とを定義するだけでとけるばあいがおおい．このようにできるだけ少数の規則 (単純なプログラム) で問題をとこうというかんがえかたは，AI でつかわれてきた従来のプロダクション・システムにはなかったとかんがえられる．

けが問題になる．図 3.6 のばあいには交換によって局所秩序度の和が増加するので規則を適用する．

上記のように交換する 2 個のクウィーンのあいだの関係だけをかんがえるかぎりは局所秩序度は変化しない．そのために，この規則に第 3 のクウィーンを登場させる必要が生じている．



適当な 3 個のクウィーンのあいだで規則適用前と適用後における局所秩序度を計算する。 → 局所秩序度が増加するときだけ，それらのクウィーンに対して実際に規則を適用する．

図 3.6　*N* クウィーン問題のプログラムにおける局所秩序度の計算

4. CPM によるプログラムのふるまい

この章では CPM によって記述されたプログラムの探索空間をまず 4.1 節で静的に分析し，また 4.2 節と 4.3 節とでその動的なふるまいについてのべる．いずれも直観にもとづく議論であって理論的なうらづけがえられていない部分がおおいことを，あらかじめことわっておく．また，ページ数がかぎられているので，いずれも概要をのべるにとどめる．

4.1 探索空間とオペレータ

作業記憶がとりうるすべての状態からなる離散的な空間をかんがえることができる．CPM におけるプログラムの実行とは，この空間のなかを移動しながら解をあらわす点を探索することである．したがって，この空間を探索空間とよぶ．インスタンスを発火させることによって探索空間中を移動するから，インスタンスを抽象したものが AI 的な探索の理論でいう“オペレータ”である．探索空間においてオペレータ (規則) の 1 回の適用で移動できる点が探索空間における隣接点であると定義する．逐次処理を仮定すると，プログラムの実行のトレースはオペレータの列によって表現することができる．

初期状態から到達不能な点を探索空間からのぞくとすれば，原子の個数がふえないシステムにおいては探索空間は有限集合になりオペレータも有限個になるから，実行前にすべてのオペレータを生成できる．*N* クウィーン問題においては原子の個数はかわらないから上記の条件をみたし，オペレータは互換をあらわすから，トレースの集合は交代群となる[注29]．

通常，CPM にもとづいて記述されたプログラムの探索空間は，多数の隣接点をもつ頂点からなる網 (ネットワーク) 構造になるとかんがえられる．この点で，*N* クウィーン問題の探索空間は典型的である．5 章でしめすように，CPM にもとづいて記述されたプログラムは非常に単純であるにもかかわらず，バックトラックをつかったプログラムなどにくらべて計算のオーダがひくいばあいがあることが経験的にわかっているが，この現象は上記のような探索空間の構造によるものではないかとおもわれる．

すなわち，第 1 にバックトラックにもとづく探索では探索空間の各頂点の隣接点数がすくないため，解へのみちのりがながい．そのために探索に時間がかかるのではないだろうか．CPM における探索ではその逆である．隣接点の個数は“自由度” (非決定性) のおおきさだとかんがえられる．したがって，探索空間の自由度をある程度おおきくしたほうが探索効率をあげられるといえるのではないだろうか[注30]．

第 2 に，バックトラックにもとづく探索では探索空間が木構造であるため，せっかく解にちかい状態に達してもバックトラックによってそこからはなれてしまい，探索に時間がかかるのではないだろうか．

4.2 安定性と適応性

CPM にもとづくシステムは動力学系として表現できる．システムの動作は非決定的だから，それは確率的な動力学系として表現される．CPM にもとづくプログラムを実行させ，ある時刻以降は入力がなくなったと仮定すると，その終状態はつぎのうちのいずれかになる．

(1) 停止 : どのインスタンスをとっても規則の適用によって秩序度が増加しない (または減少する) 状態にいたると，実行は停止する．

(2) リミット・サイクル (無限ループ 1) : 作業記憶の状態が周期的に変化する．

(3) 発散 (無限ループ 2) : 作業記憶内につぎつぎに原子が生成されるため，停止せず，リミット・サイ

[注29] CPM の理論においては，このようなオペレータとトレースの代数や確率的な動力学の理論にもとづいてシステムを数理的に検証できるようにしたいとかんがえている．

[注30] CPM においては，プログラムが複数の規則を合成してあらたな規則をつくるというような学習をつうじて自由度を獲得するような機構をかんがえることもできる．

クルにもおちいらない．

当然のことながら，規則が原子を生成しないプログラムにおいては (3) の現象は生じえない．連続空間の動力学系においては (1), (2) のほかにカオスにおちいるという現象が生じうるが，CPM における作業記憶は離散空間であるからカオスは生じない[注31]．

探索空間が網構造をした自由度のたかいシステムは，環境 (あらたな入力) への適応性 (自己組織性) があるのではないかとかんがえられる．すなわち，自由度があるとシステムの構成要素が自発的に動作して適応するという行動をとりやすいのではないだろうか．また，近接した状態をたどっていくことによって，バックトラックのような "不連続的な" おおきなジャンプなしにあたらしい安定状態に移行できるのではないだろうか．このようなシステムは多数の局所安定状態をもつとかんがえられるから，安定状態からはずれてもその近傍に他の安定状態をみつけやすく，破壊的なおおきなジャンプをへる必要がないのではないかとかんがえられる．

つぎに，バグに対する "安定性" についてのべる．従来の方法で記述されたプログラムにおいては，微小なバグが結果に重大な影響をおよぼしうる．それは，プログラムが "目標" (または目的) をしらないから，わずかな撹乱すなわちバグによって目標 からかけはなれてしまうのだと解釈できる．それに対して，CPM のプログラムにおいては，秩序度関数の定義がまちがっていなければ，規則のなかに意図しない部分があったとしてもただしい解をもとめられるばあいがおおいとかんがえられる．なぜなら，システムには秩序度関数という目標が陽にあたえられているため，撹乱によってそれがみうしなわれにくいとかんがえられるからである．

いうまでもなく，秩序度関数の定義域に関してバグがあれば結果に重大な影響があることはさけられない．しかし，秩序度関数の値のほうは撹乱につよいとかんがえられる．たとえば，前記の N クウィーン問題のプログラムにおける秩序度の値 1 を 2 や 10 にかえてもプログラムの動作にはなんの影響もない．

4.3 スケジュリング戦略と非合理性の導入

CPM の実行過程において，同時に発火させることができるインスタンスのうち，どのインスタンスをどのような順序で発火させるかをきめるのがスケジュリング戦略である．スケジュリング戦略ということばは，従来のプロダクション・システムにおける競合解消戦略にちかい意味をもっているが，よりひろい意味をもたせるために，ことなる用語をつかうことにした[注32]．スケジュリング戦略は，CPM に特有の非決定性，競合にもとづく組織化にかかわっている．したがって，秩序度関数とともに CPM のもっとも重要な部分だとかんがえられる．

CPM におけるスケジュリング戦略としては決定論的なものもかんがえられるが，一般的には非決定論的である．したがって，スケジュリングのための手段として，つぎのようなものがかんがえられる．

(1) 決定論的な戦略 : 系統的な方法でインスタンスを選択する戦略．

(2) 偶然性にもとづく戦略 : 乱数などによってランダムにインスタンスを選択する戦略．

(3) 人間による主体的・主観的選択戦略 : 人間の直観的な判断，感情的な判断などにもとづいてインスタンスを選択する戦略．

(4) 超越的な存在による選択戦略 : 自然現象 (たとえばある種のカオス現象) をつかったインスタンスの非合理的選択，人間による非主体的選択，あるいは "神" (またはシャーマン) による選択などにもとづく戦略．

これらのうち (1) は合理性がつよいとかんがえられるが，(3), (4) は非合理性がつよい．自然現象を利用するとしても，その理論的な背景がわかっているばあいは (1) に分類できるが，それがわからないばあいは (4) に分類できるであろう．並列発火をみとめる合理的戦略は，(1) と (2) の混合戦略になるだろう．このような並列戦略については後述する．

これらの手段による戦略を，以下，順に説明する．

(1) 決定論的な戦略

決定論的な戦略としては，インスタンスを秩序度の増加がおおきい順にならべて先頭から実行するという，従来のプロダクション・システムの競合解消にちかい戦略もありうる．しかし，これは適当な戦略ではないであろう．なぜなら，この戦略はむだな計算 (つかわないインスタンスの計算) がおおく，また並列性を阻害するからである．

上記の戦略以外にも決定論的な戦略としてはさまざまなものがかんがえられる．5 章においては

[注31] ただし，(3) のばあいのなかにはカオスにちかい現象がおこるばあいがあるといえるかもしれない．

[注32] 競合解消とよばないひとつの理由は，自己組織化においては競合が重要なやくわりをはたすから，むしろ積極的にそれを発生させることが自己組織化につながるばあいがあるとかんがえられるからである．

N クウィーン問題の実行のために "最右条件優先戦略" を使用するが，これもそのひとつである[注33]．

(2) 偶然性にもとづく戦略

ランダム戦略においては，規則，各条件にマッチするデータのすべてがランダムに選択される．さまざまな分布のランダム戦略がかんがえられる．

(3) 人間による主体的・主観的選択

人間による選択にもとづく戦略においては，規則を適用するごとに人間による選択が必要だとするといちじるしく推論速度を減速する．したがって，通常は秩序度関数に人間の判断をプログラムしておくことになるだろう．この方法は，ファジィ推論においてあらかじめメンバシップ関数のかたちを人間が主観的にきめておく方法と似ている．

(4) 超越的な存在による選択

この方法は自然システムにおける "神の手" を人工システムにそのまま利用しようというものである．自然の自己組織系における "神の手" による選択がどのような戦略にもとづいているかを人間が知ることはほとんど不可能である．自己組織系を統計的手法で解析することはできるが，統計的手法で "神の手" がうまくとらえられるとはいえない[注34]．それならば "神の手" をそのまま利用してしまおうというのが，この方法である．

以上のような戦略のなかでいずれがよりよいかを理論的にきめるのはむずかしい．とくに非合理的な戦略についてはそうである．したがって，さまざまなスケジュリング戦略を実験的にためしてよいものをみつける必要があるとかんがえられる．

つぎに，並列スケジュリング戦略についてのべる．CPM においては，化学反応系と同様の並列性の実現を可能にしようとかんがえている．このかんがえかたは，Berry らの Chemical Abstract Machine [Ber 90] と似ている．モデルも秩序度関数をのぞけば似ている．並列性のあるスケジュリング戦略としては，発火により秩序度が増加する，原子にかさなりのないインスタンスをすべて並行して発火させるという戦略が単純かつのぞましいであろう．CPM は非常に粒度がちいさい並列処理モデルとなっているから，超並列 (Massively Parallel) 計算機においてうまく並列実行できるのではないかとかんがえている．

最後に，スケジュリング戦略に対するプログラムの感度に関する問題についてのべる．従来のプロダクション・システムのプログラムにおいてよくおこなわれるように，陽にフロー制御用の原子を導入することにより決定論的なプログラムを記述したばあいには，スケジュリング戦略によってその動作が影響されない．非決定論的なプログラムのなかにはスケジュリング戦略に対して高感度のもの (すなわち非決定性がたかいもの — 競合性がたかいもの) と低感度のもの (非決定性がひくいもの) とがある．たとえば，*N* クウィーンのプログラムは高感度だが，巡回セールスマン問題のプログラムは低感度だとわかっている．スケジュリング感度がたかいプログラムのほうがより "化学的" だとかんがえられる．しかし，感度が極端にたかいプログラムは適当なスケジュリング戦略をみつけるのがきわめて困難であり，実用的なプログラムではないとかんがえられる．

[注33] 決定論的な戦略においては，発火させるインスタンスを (1) 規則と条件のいずれを優先してきめるか，(2) 規則にふくまれるマッチング条件のうちのどれを優先してきめるか，さらに (3) 作業記憶内の原子のうちのどれを優先してきめるかなどに関してそれぞれ選択枝があり，これらのくみあわせとしてさまざまな戦略が定義される．これらのうち，あとで必要になる (2) についてだけ説明する．

(2) に関する選択枝として，もっとも右側の条件をよりはやくかえながらテストする最右条件優先戦略と，その逆の最左条件優先戦略とがある．*N* クウィーン問題のばあいについていえば，最右条件優先戦略によるプログラムの実行は，つぎのような 3 重ループの実行にちかい (わかりやすくするために単純化をはかっているので，完全にここにしめしたとおりというわけではない．しかし，*N* クウィーン問題のプログラムにおいては規則はただひとつなのでそれをスケジュリングにおいて考慮する必要がないという点は，このプログラムのとおりである)．なお，ここで〈Swap, Queen1, Queen2, Queen3〉は規則が swap, 条件にマッチするクウィーンが左から Queen1, Queen2, Queen3 であるインスタンスをあらわす．

```
for Queen1 in WorkingMemory do
    for Queen2 in WorkingMemory −{Queen1} do
        for Queen3 in WorkingMemory −{Queen1, Queen2} do
            if 〈Swap, Queen1, Queen2, Queen3〉
                が発火可能 then
                発火;
            end if; end for; end for; end for;
```

(1), (2), (3) においてどの選択枝をとるときも，ひとつの原子や規則をつかってからつぎにそれらをつかうまでのあいだに他のすべて (あるいはほとんど) の原子や規則をみるという意味での公平性があることが重要だとかんがえられる．公平性がないとうまく動作しないプログラムがあることがわかっている．

[注34] なぜなら，統計的手法は複雑なシステムを人間に理解できるレベルまで単純化をはかるために，情報のきりすてあるいは圧縮をおこなっているからである．統計解析結果からもとのシステムを復元することは，よほど性質のよいシステムでなければできない．したがって，"神の手" をまねて人間が戦略をきめても，自己組織系が実現できるという保証はない．ただし，Haken らはこのような "復元" がひろい範囲で可能だとかんがえているようである．

5. CPM による小規模問題の実験

CPM にもとづくプログラミング言語 SOOP[注35] とその処理系を開発し，いくつかの小規模問題のプログラミングと実行をこころみた．これらのなかで，5.1 節では N クウィーン問題のプログラムの実験結果，5.2 節にそれ以外の実験結果の要約をしめす．

5.1 N クウィーン問題の実験

前述の N クウィーン問題のプログラムを SOOP によって記述し，実験した．その結果を要約する．インスタンス間の競合のため，決定論的なスケジュリング戦略を使用するとリミット・サイクルにおちいるばあいもあるが，ランダムに生成した初期状態から計算をはじめても大半は停止することがわかった[注36]．また，解以外の状態で停止することはない．

最右条件優先戦略をつかって停止するばあいについて，解がもとまるまでの計算時間を $N = 50$ までもとめたところ，ほぼ $O(N^4)$ であることがわかった．計算時間は規則の左辺をしらべた回数にほぼ比例する．バックトラックをつかって解を計算するには指数的な時間がかかるから，このプログラムはそれよりはるかにはやい．ただし，$O(N)$ で解を生成する方法 [Aki 91] にくらべればはるかにおそい．

図 5.1 にはリミット・サイクルにおちいらずに解がもとめられる確率を 100 回の試行から実験的にもとめてしめした．縦軸は確率を対数スケールであらわしていて，上限が 1 となっている．

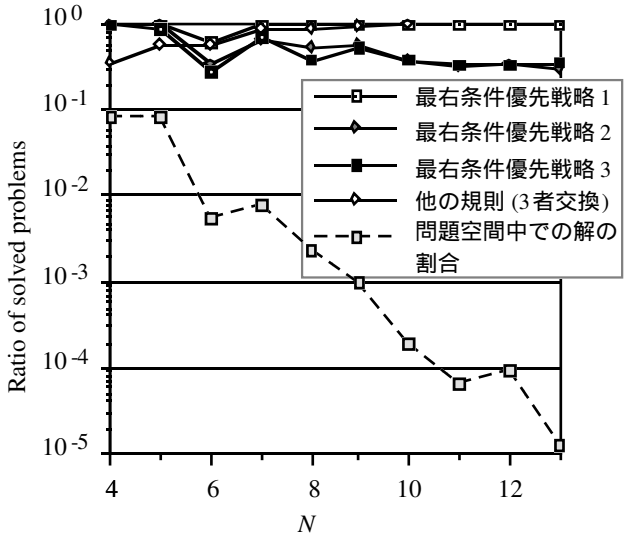


図 5.1 N クウィーンにおける解がもとめられる確率

この図には，前記のプログラムにおける条件の出現順をかえて最右条件優先戦略のもとでもとめた確率 (スケジュリング戦略 1 ~ 3) と，前記のとはことなる規則 (3 者交換規則) をつかってもとめた確率とをあわせてしめした．初期配置はランダムにきめた．また，この図にはクウィーンのランダムな配置を生成したときにそれが解になっている確率 (問題空間中での解の割合) をもあわせてしめした．

この図から，解がもとめられる確率のスケジュリング戦略に対する感度がたかいことがわかる．すなわち，スケジュリング戦略 1 においては $N = 6$ のときをのぞいてほぼ確率 1 で解がもとめられる．他の戦略はこれよりはるかにおとっているが，配置をランダムに発生させるのにくらべるとはるかによい．この結果は，CPM におけるスケジュリング戦略の重要性を示唆しているとかんがえられる．

つぎに，実験などからわかった N クウィーン問題のプログラムの 2 つの性質についてのべる．

(1) 漸進性

漸進性は CPM によるプログラムの一般的な特徴であるが，このプログラムのばあいは，つぎのような漸進性がある．N クウィーン問題の解がえられたところで $N + 1$ 番めのクウィーンをシステムの外部たとえばユーザからあたえると，$N + 1$ クウィーン問題の解がもとめられる．

(2) 探索における極大点からのぬけだし

このプログラムにおいては，探索を局所的な情報にもとづいて分散的かつ競合的におこなうことによって，大域的な情報による集中的な探索においてはさけることが困難な袋小路へのおちこみを回避しているといえるだろう．すなわち，大域秩序度が極大になる点においても局所秩序度が増加するようなクウィーンのくみあわせが存在するため，極大点をぬけだすことができる．そして，それゆえに決定論的な戦略においても解がもとめられる確率が向上している．このプログラムにおいては，N クウィーン問題においては盤面を分割統治できないという性質がむしろ有利にはたらいているようにおもわれる．このような現象がもし広範におこりうるなら，非還元的な問題解決の方法へと発展させていけるのではないだろうか．

上記の N クウィーン問題のプログラムにおいては，その解決手順を構成するためのステップを記述するだけで問題解決が可能になっている．したがって，1 章でのべたような問題解決手順の自己組織化がお

[注35] SOOP = Self-Organization-Oriented Programming. [soup] ではなく [su: p] と発音する．

[注36] インスタンスの選択にランダムネスを導入すれば，規則的な選択によってはリミット・サイクルにおちいるばあいでも，それをふせぐことができる．

こなわれているともかんがえられる．しかし，このプログラムは本来の自己組織系がもつべき性質の一部をもつだけであり，自己組織系だとはいえないであろう．また，このプログラムの動作に関してはまだわからないことがおおく，その解析は今後の課題である．たとえば，なぜ停止しないばあいがあるかという問に対するこたえはまだえられていない．

5.2 N クウィーン問題以外の実験結果

上記の N クウィーン問題のほかに，これまでに巡回セールスマン問題 (TSP)，輸送問題 ,ニューラル・ネットによる N クウィーン問題などを SOOP によって記述して実験した．これらのうちTSP は，10× 10 の格子点に 8 個または 9 個の都市を配置すると，いずれのばあいも10 回の試行のうち 9 回は最適解がもとめられた．また，都市数 500 まで実験をおこない，計算時間はほぼ $O(N^3)$ であることがわかった．TSP においても，解がもとめられたあとで都市を追加または削除することにより再動作させられる．

6. 結言

この報告では，自己組織系がみたすべき必要条件のいくつかをしめし，それをみたせる計算モデル CPM を提案し分析した．また，CPM によって N クウィーン問題などのくみあわせ問題の柔軟なプログラムが非常に簡潔に記述でき，非常に効率的に実行されることをしめした．CPM を発展させれば，モデルのなかに人間のもつ不完全性や非合理性までをもとりこみつつ，数理解析が可能なシステムをつくれるのではないかとおもわれる．また，分割統治法がうまくはたらかない全体性や開放性をもつ問題があつかえるようになると期待される．まだ自己組織化の本質に十分に接近できたとはいえないが，この研究はコンピュータによる自己組織系の実現のための一歩となるとかんがえられる．

今後の課題のなかから，重要な 2 点をあげる．第 1 に，N クウィーン問題や TSP のプログラムは基本的に閉鎖系であり，自己組織性が本来のちからを発揮する問題ではないとかんがえられる．今後，より適切な問題に CPM を適用し，その結果をモデルに反映させていくことが必要であろう．

第 2 に，現在の CPM においては秩序度関数すなわち計算の目標や目的を絶対のものとしている．しかし，本来，自己組織化の過程では，計算によってえられたあらたな知識にもとづいて目標や目的が修正されるばあいがおおいであろう．そこで，モデルを拡張してそれに対処する必要があるとかんがえられる．これに関連して，CPM への自己参照 (リフレクション) の導入方法をかんがえる必要があろう．

参考文献

[Aki 91] 秋葉 澄孝 : N-クウィーン問題の解を線形時間で求めるアルゴリズムについて, *情報処理学会第 42 回全国大会*, 5B-6, 1991.

[Ber 90] Berry, G., and Boudol, G.: The Chemical Abstract Machine, *Proc. 17th Annual ACM Symposium on Principles of Programming Languages*, pp. 81–94, 1990.

[Deg 88] 出口 弘 : 自己組織化研究の方法論批判, *現代思想*, Vol. 16, No. 1, pp. 128–137, 1988.

[For 81] Forgy, C. L.: *OPS5 User's Manual*, Technical Report CMU-CS-81-135, Carnegie Mellon University, Dept. of Computer Science, 1981.

[Hak 78] H. ハーケン : *協同現象の数理* (小森・相沢 訳), 東海大学出版会, 1980.

[Hak 81] H. ハーケン : *自然の造形と社会の秩序* (高木 隆司 訳), 東海大学出版会, 1985.

[Hak 83] H. ハーケン : *シナジェティクスの基礎* (小森・相沢 訳), 東海大学出版会, 1986.

[Iga 80] 五十嵐 善英 : 確率的アルゴリズムの概観, *情報処理*, Vol. 21, No. 1, pp. 13–18, 1980.

[Jan 80] E. ヤンツ : *自己組織化する宇宙* (芹沢 高志・内田 美恵 訳), 工作舎, 1986.

[Las 72] E. ラズロー : *システム哲学入門* (伊藤 重行 訳), 紀伊國屋書店, 1980.

[Nis 88] 西垣 通 : *AI — 人工知能のコンセプト*, 講談社現代新書, 1988.

[Nis 90] 西垣 通 : 秘術としての *AI 思考*, 筑摩ライブラリー, 1990.

[Pri 77] G. ニコリス, I. プリゴジーヌ : *散逸構造* (小畠・相沢 訳), 岩波書店, 1980.

[Pri 84] I. プリゴジン, I. スタンジェール : *混沌からの秩序* (伏見 康治 他訳), みすず書房, 1987.

[Sau 49] F. ソシュール : *一般言語学講義* (小林 英夫 訳), 岩波書店, 1940.

[Shi 87] 清水 博 他 : *バイオ情報システムに関する調査報告書*, 62-A-256, 日本電子工業振興協会,1987.

[Shi 88] 清水 博 他 : *バイオ情報システムに関する調査報告書*, 63-A-298,日本電子工業振興協会, 1988.

[Shi 90] 清水 秀夫, 林 彬 : n-Queen 問題の高速解法, *電気情報通信学会技術報告*, COMP 90-39, pp. 79–86, 1990.

[Sim 81] H. A. サイモン : *システムの科学* (稲葉・吉原 訳), パーソナルメディア, 1987.

[Tak 86] 今田 高俊 : *自己組織性 — 社会理論の復活 —*, 創文社, 1986.

[Tak 91] 竹内 郁雄, 後藤 滋樹, 尾内 理紀夫, 斎藤 康己, 奥野 博 : *コネクティクス構想*, 日本ソフトウェア科学会第 8 回大会論文集, B3-1, 1991.

[Tsu 88] 辻井 潤一, 安西 祐一郎 : *機械の知 人間の知*, 認知科学選書 20, 東京大学出版会, 1988.